Algorithms and Data structures - (Lab 4)

Submitted to:

Prof. Dr. Ulrike Herster

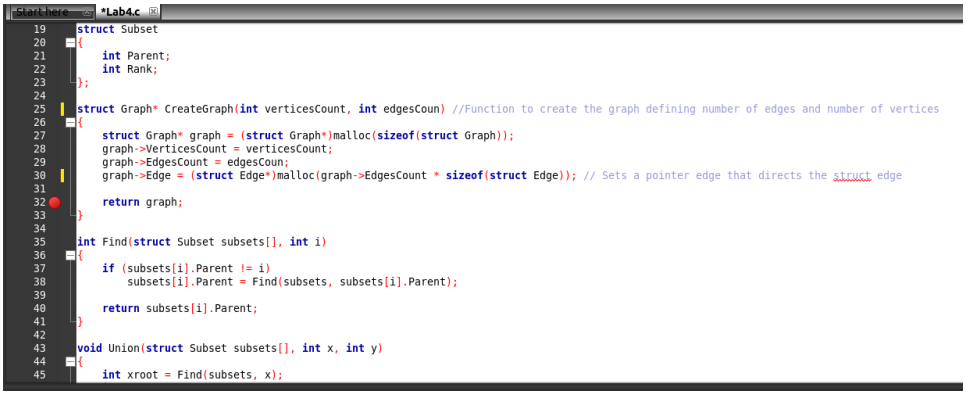
Submitted by:

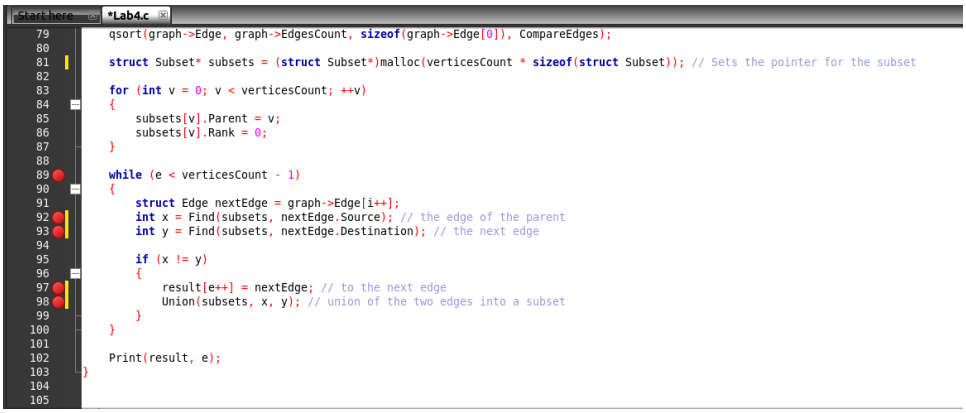
- MInhazul Islam
- MD Moniruzzaman

Assignment 1: Kruskal's algorithm for calculating the minimal spanning tree:

1.1 Understanding the program:

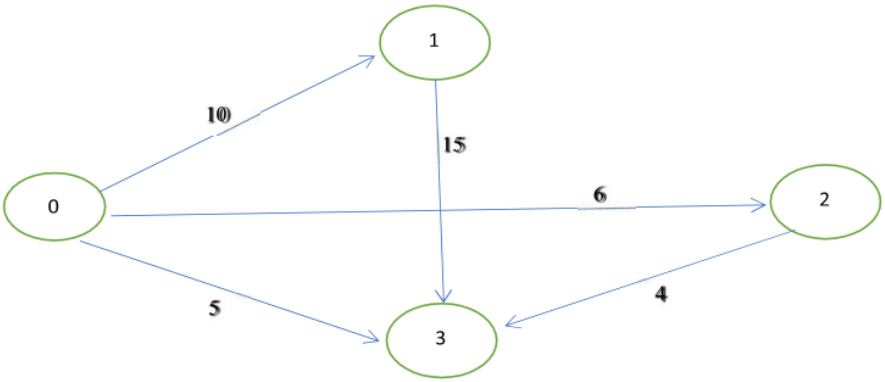
In this implementation, a graph data structure is utilized to facilitate the discovery of the minimum spanning tree. Three key structures are employed: Edge, Graph, and Subset. The essential functions include CreateGraph (for graph initialization with specified edges and vertices), Find (to identify the parent of a given subset), Union (for merging two edges into a subset), CompareEdges (for comparing the weights of two edges), and Kruskal's algorithm. The initial demonstration showcases the creation of a graph with 4 vertices and 5 edges. Each edge is associated with a pointer indicating its weight and source. Moving on to the Kruskal function, a subset pointer is allocated. The edges are sorted quickly, and during selection, they are compared to determine their parent and subsequently united into a subset. This process iterates through the edges (i++) until all vertices in the graph are accounted for, resulting in the identification of the shortest spanning tree.

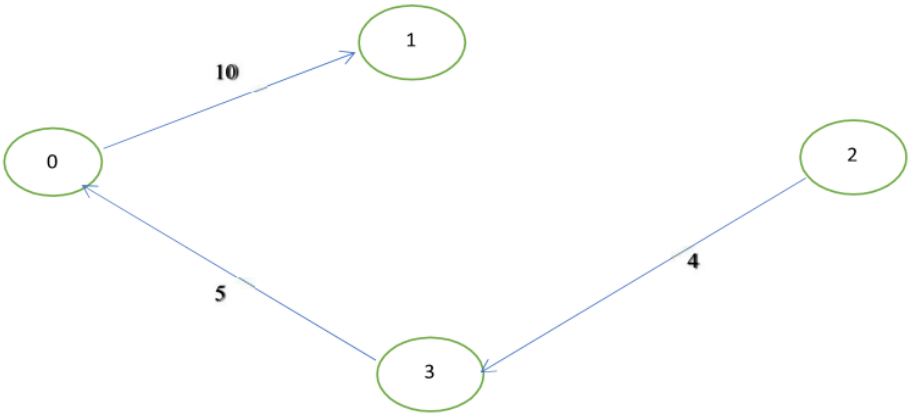




Assignment 1.2: Understanding the program

Two diagrams have been created here. The initial diagram corresponds to the main function code, while the second one represents the minimum spanning tree. The identified minimum spanning tree follows the path 2 -> 3 -> 0 -> 1, with a total weight of 19.





Assignment 1.3: Program evaluation

Here are three things that I like about Kruskal's algorithm:

(1)The efficiency of Kruskal's algorithm is enhanced by the utilization of the quick sort function.

(2)Another notable advantage is its simplicity, making it easy to use and understand.

(3)This algorithm is applicable to various types of graphs, encompassing both directed and undirected graph structures..

Here are two things that I don't like about Kruskal's algorithm:

(1)Prim's algorithm exclusively explores neighboring nodes for each node, while Kruskal's algorithm necessitates the sorting of edges. This distinction contributes to Kruskal's algorithm being less efficient in dense graphs.

(2)Prim's algorithm has less complexity than the Kruskal algorithm.

1.4 Line 75 From my point of view, line 75 is not correct. What do you think bothers me about this line? How would you rewrite this line?

In the line number 75:

struct Edge* result = (struct Edge*)malloc(sizeof(result) * verticesCount); //Error?

Here it should be (verticesCount –1), because the number of verticesCount should be smaller by 1 count from the edge count.

Assignment 2.1 Representation:

Adjacency list:

A->B,3->D,2->C,2/

B->E,7->D,5/ C->F,6/

D->E,5->C,4->F,12/

E->G,5/

F->G,2/

G/

Adjacency Matrix:

	A	B	C	D	E	F	G
A	0	3	2	2	0	0	0
B	0	0	0	5	7	0	0
C	0	0	0	0	0	6	0
D	0	0	4	0	5	12	0
F	0	0	0	0	0	0	2
E	0	0	0	0	0	0	5
G	0	0	0	0	0	0	0

Assignment 2.2: dfs and bfs:

Depth first Search:

Here, only the unvisited vertex is visited and if it ends, we go back to a vertex which allows us to visit a new one. From A to G, we are moving according to the smallest weight and then from G we go back to vertex D and then to E and then only B is left. We are using a stack to keep track of the visited vertexes.

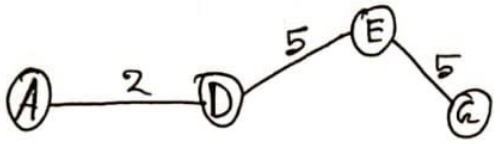
A->D->C->F->G->E->B

Breadth first Search:

Here, a Queue is used, and we visit all the vertices in the order of their weight: D->B->C and then removed 1 from the que and then visited D. vertex D connect us to E->F. Then the only unvisited vertex left is G.

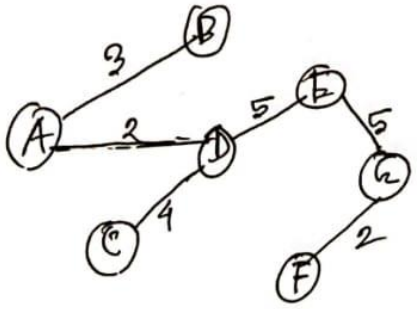
A->D->B->C->E->F->G

Assignment 2:3: Dijkstra's Algorithm



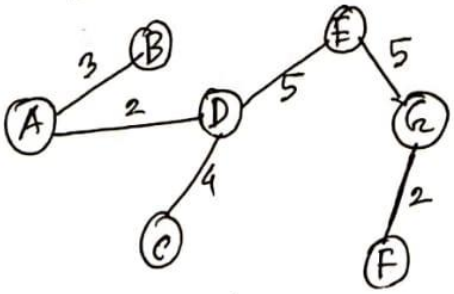
$A \rightarrow D \rightarrow E \rightarrow G = 12$

Assignment 2.4: Kruskal's Algorith:



$\therefore$ MST = 21.

Prim's Algorith



$\therefore$ MST = 21.